

Python: module cdms.dataset

cdms.dataset

[index](#)

CDMS dataset and file objects

Modules

cdms.Cdunif	cdms.cdmsNode	cdms.internattr	sys
MA	cdms.cdmsURLopener	os	urllib
Numeric	cdms.cdmsobj	re	urlparse
cdms.cache	cdms.convention	string	

Classes

[cdms.cdmsobj.CdmsObj\(cdms.internattr.InternalAttributesClass\)](#)
[CdmsFile\(cdms.cdmsobj.CdmsObj, cdms.cudsinterface.cuDataset\)](#)
[Dataset\(cdms.cdmsobj.CdmsObj, cdms.cudsinterface.cuDataset\)](#)
[cdms.cudsinterface.cuDataset](#)
[CdmsFile\(cdms.cdmsobj.CdmsObj, cdms.cudsinterface.cuDataset\)](#)
[Dataset\(cdms.cdmsobj.CdmsObj, cdms.cudsinterface.cuDataset\)](#)
[cdms.error.CDMSError\(exceptions.Exception\)](#)
[DuplicateAxisError](#)

class **[CdmsFile\(cdms.cdmsobj.CdmsObj, cdms.cudsinterface.cuDataset\)](#)**

Method resolution order:

[CdmsFile](#)
[cdms.cdmsobj.CdmsObj](#)
[cdms.internattr.InternalAttributesClass](#)
[PropertiedClasses.Properties.PropertiedClass](#)
[cdms.cudsinterface.cuDataset](#)

Methods defined here:

```
\_\_delattr\_\_\(self, name\)
    # delattr deletes external global attributes in the file

\_\_getattr\_\_\(self, name\)
    # getattr reads external global attributes from the file

\_\_init\_\_\(self, path, mode\)

\_\_repr\_\_\(self\)
```

```

setattr(self, name, value)
    # setattr writes external global attributes to the file

close(self)

copyAxis(self, axis, newname=None, unlimited=0, index=None, extbounds=None)
    # Copy axis description and data from another axis

copyGrid(self, grid, newname=None)
    # Copy grid

createAxis(self, name, ar, unlimited=0)
    # Create an axis
    # 'name' is the string name of the Axis
    # 'ar' is the 1-D data array, or None for an unlimited axis
    # Set unlimited to true to designate the axis as unlimited
    # Return an axis object.

createRectGrid(self, id, lat, lon, order, type='generic', mask=None)
    # Create an implicit rectilinear grid. lat, lon, and mask are
    # order and type are strings

createVariable(self, name, datatype, axesOrGrids, fill_value=None)
    # Create a variable
    # 'name' is the string name of the Variable
    # 'datatype' is a CDMS datatype or Numeric typecode
    # 'axesOrGrids' is a list of axes, grids. (Note: this should
    #     generalized to allow subintervals of axes and/or grids)
    # Return a variable object.

createVariableCopy(self, var, id=None, attributes=None, axes=None, extbounds=None, extend=0,
newname=None, grid=None)
    Define a new variable, with the same axes and attributes as var.
    This does not copy the data itself.
    Keywords:
        attributes: A dictionary of attributes. Default is var.attributes
        axes: The list of axis objects. Default is var.getAxisList()
        extbounds: Bounds of the (portion of) the extended dimension
        id or newname: String identifier of the new variable.
        extend: If 1, define the first dimension as the unlimited dimension
            an unlimited dimension. The default is the define the first
            only if it is a time dimension.
    - fill_value is the missing value flag.
    - index is the extended dimension index to write to. The default
        by lookup relative to the existing extended dimension.
    grid is the variable grid. If none, the value of var.getGrid()

createVirtualAxis(self, name, axislen)
    Create an axis without any associated coordinate array. This
    axis is read-only. This is useful for the 'bound' axis.
    <name> is the string name of the axis.
    <axislen> is the integer length of the axis.

```

Note: for netCDF output, this just creates a dimension without the associated coordinate array. On reads the axis will look an axis of type float with values [0.0, 1.0, ..., float(axis)]. On write attempts an exception is raised.

getAxis(self, id)

Get the axis object with the given id. Returns None if not found.

getBoundsAxis(self, n)

Get a bounds axis of length n. Create the bounds axis if necessary.

getGrid(self, id)

Get the grid object with the given id. Returns None if not found.

getVariable(self, id)

Get the variable object with the given id. Returns None if not found.

getVariables(self, spatial=0)

Get a list of variable objects. If spatial=1, only return the axes defined on latitude or longitude, excluding weights and bounds.

matchPattern(self, pattern, attribute, tag)

```
# Match a pattern in a string-valued attribute. If attribute
# search all string attributes. If tag is 'cdmsFile', just check
# else check all nodes in the dataset of class type matching
# is None, search the dataset and all objects contained in it.
```

searchPattern(self, pattern, attribute, tag)

```
# Search for a pattern in a string-valued attribute. If attribute
# search all string attributes. If tag is 'cdmsFile', just check
# else check all nodes in the dataset of class type matching
# is None, search the dataset and all objects contained in it.
```

searchPredicate(self, predicate, tag)

```
# Apply a predicate, returning a list of all objects in the dataset
# for which the predicate is true. The predicate is a function that
# takes a dataset as an argument, and returns true or false.
# tag is 'cdmsFile', the predicate is applied to the dataset
# If 'variable', 'axis', etc., it is applied only to that type of
# object in the dataset. If None, it is applied to all objects, including
# the dataset itself.
```

sync(self)

write(self, var, attributes=None, axes=None, extbounds=None, id=None, extend=None, fill_value=None)

Write var to the file. If the variable is not yet defined in the definition is created. By default, the time dimension of the 'extended dimension' of the file. The function returns the coordinates.

Keywords:

- attributes is the attribute dictionary for the variable.

- axes is the list of file axes comprising the domain of the variable.
copy var.getAxisList().
- extbounds is the extended dimension bounds. Defaults to None.
- id is the variable name in the file. Default is var.id.
- extend=1 causes the first dimension to be 'extensible': it is 'extensible' if its size is greater than 1. The default is None, in which case the first dimension is not extensible. Set to 0 to turn off this behaviour.
- fill_value is the missing value flag.
- index is the extended dimension index to write to. The default is None, in which case the value is written by lookup relative to the existing extended dimension.

Methods inherited from [cdms.cdmsobj.CdmsObj](#):

dump(self, path=None, format=1)

dump(self, path=None, format=1)

Dump an XML representation of this object to a file.
'path' is the result file name, None for standard output.
'format'==1 iff the file is formatted with newlines for readability.

matchone(self, pattern, atname)

Return true iff the attribute with name atname is a string
attribute which matches the compiled regular expression pattern.
if atname is None and pattern matches at least one string
attribute. Return false if the attribute is not found or is not a string.

searchone(self, pattern, atname)

Return true iff the attribute with name atname is a string attribute which contains the compiled regular expression pattern. If atname is None and pattern matches at least one string attribute. Return false if the attribute is not found or is not a string.

Methods inherited from [cdms.internalattr.InternalAttributesClass](#):

is_internal_attribute(self, name)

is internal attribute(name) is true if name is internal.

replace_external_attributes(self, newAttributes)

replace external attributes(newAttributes)

Replace the external attributes with dictionary newAttributes.

Methods inherited from [PropertiedClasses.Properties.PropertiedClass](#):

get_property_d(self, name)

Return the 'del' property handler for name that self uses.
Returns None if no handler.

get_property_g(self, name)

Return the 'get' property handler for name that self uses.
Returns None if no handler.

```
get_property_s(self, name)
    Return the 'set' property handler for name that self uses.
    Returns None if no handler.

set_property(self, name, actg=None, acts=None, actd=None, nowrite=None, nodelete=None)
    Set attribute handlers for name to methods actg, acts, actd
    None means no change for that action.
    nowrite = 1 prevents setting this attribute.
        nowrite defaults to 0.
    nodelete = 1 prevents deleting this attribute.
        nodelete defaults to 1 unless actd given.
    if nowrite and nodelete is None: nodelete = 1
```

Methods inherited from [cdms.cudsinterface.cuDataset](#):

```
__call__(self, id, *args, **kwargs)
    Call a variable object with the given id. Exception if not fo
    Call the variable with the other arguments.

__getitem__(self, key)
    Implement f['varname'] for file/dataset f.

cleardefault(self)
    Clear the default variable name.

default_variable(self, vname)
    Set the default variable name.

dimensionarray(self, dname, vname=None)
    Values of the dimension named dname.

dimensionobject(self, dname, vname=None)
    CDMS axis object for the dimension named dname.

getattribute(self, vname, attribute)
    Get the value of attribute for variable vname

getdimensionunits(self, dname, vname=None)
    Get the units for the given dimension.

getglobal(self, attribute)
    Get the value of the global attribute.

getslab(self, vname, *args, **keys)
    getslab('name', arg1, arg2, ....) returns a cdms variable
    containing the data.

    Arguments for each dimension can be:
    (1) : or None -- selected entire dimension
    (2) Ellipsis -- select entire dimensions between the ones
    (3) a pair of successive arguments giving an interval in
        world coordinates.
```

(4) a cdms-style tuple of world coordinates e.g. (start, stop, step)

listall(self, vname=None, all=None)
Get info about data from the file.

listattribute(self, vname=None)
Get attributes of data from the file.

listdimension(self, vname=None)
Return a list of the dimension names associated with a variable.
If no argument, return the file.axes.keys()

listglobal(self)
Returns a list of the global attributes in the file.

listvariable(self)
Return a list of the variables in the file.

listvariables = *listvariable*(self)
Return a list of the variables in the file.

readScripGrid(self, whichGrid='destination', checkGrid=1)
Read a SCRIP curvilinear or generic grid from the dataset.
The dataset can be a SCRIP grid file or mapping file. If a mapping file is used, 'whichGrid' chooses the grid to read, either "source" or "destination". If 'checkGrid' is 1 (default), the grid cells are checked for consistency and 'repaired' if necessary.
Returns the grid object.

showall(self, vname=None, all=None, device=None)
Show a full description of the variable.

showattribute(self, vname=None, device=None)
Show the attributes of vname.

showdimension(self, vname=None, device=None)
Show the dimension names associated with a variable.

showglobal(self, device=None)
Show the global attributes in the file.

showvariable(self, device=None)
Show the variables in the file.

class ***Dataset***(cdms.cdmsobj.CdmsObj, cdms.cudsinterface.cuDataset)

Method resolution order:

Dataset
cdms.cdmsobj.CdmsObj
cdms.internattr.InternalAttributesClass

[PropertiedClasses.Properties.PropertiedClass](#)
[cdms.cudsinterface.cuDataset](#)

Methods defined here:

```
__init__(self, uri, mode, datasetNode=None, parent=None, datapath=None)
__repr__(self)
close(self)
    # Close all files

createAxis(self, name, ar)
    # Create an axis
    # 'name' is the string name of the Axis
    # 'ar' is the 1-D data array, or None for an unlimited axis
    # Return an axis object.

createRectGrid(id, lat, lon, order, type='generic', mask=None)
    # Create an implicit rectilinear grid. lat, lon, and mask are
    # order and type are strings

createVariable(self, name, datatype, axisnames)
    # Create a variable
    # 'name' is the string name of the Variable
    # 'datatype' is a CDMS datatype
    # 'axisnames' is a list of axes or grids
    # Return a variable object.

getAxis(self, id)
    Get the axis object with the given id. Returns None if not fo

getConvention(self)
    Get the metadata convention associated with this dataset or f

getDictionary(self, tag)
    # Get a dictionary of objects with the given tag

getGrid(self, id)
    Get the grid object with the given id. Returns None if not fo

getLogicalCollectionDN(self, base=None)
    Return the logical collection distinguished name of this data
    If <base> is defined, append it to the lc name.

getPaths(self)
    # Return a sorted list of all data files associated with the

getVariable(self, id)
    Get the variable object with the given id. Returns None if no

getVariables(self, spatial=0)
```

Get a list of variable objects. If spatial=1, only return the axes defined on latitude or longitude, excluding weights and

matchPattern(self, pattern, attribute, tag)

```
# Match a pattern in a string-valued attribute. If attribute  
# search all string attributes. If tag is 'dataset', just che  
# else check all nodes in the dataset of class type matching  
# is None, search the dataset and all objects contained in it
```

openFile(self, filename, mode)

```
# Open a data file associated with this dataset.  
# <filename> is relative to the self.datapath  
# <mode> is the open mode.
```

searchPattern(self, pattern, attribute, tag)

```
# Search for a pattern in a string-valued attribute. If attri  
# search all string attributes. If tag is 'dataset', just che  
# else check all nodes in the dataset of class type matching  
# is None, search the dataset and all objects contained in it
```

searchPredicate(self, predicate, tag)

```
# Apply a predicate, returning a list of all objects in the da  
# for which the predicate is true. The predicate is a function  
# takes a dataset as an argument, and returns true or false.  
# tag is 'dataset', the predicate is applied to the dataset  
# If 'variable', 'axis', etc., it is applied only to that type  
# in the dataset. If None, it is applied to all objects, includ  
# the dataset itself.
```

sync(self)

```
# Synchronize writes with data/metadata files
```

Methods inherited from [cdms.cdmsobj.CdmsObj](#):

dump(self, path=None, format=1)

```
dump(self, path=None, format=1)  
Dump an XML representation of this object to a file.  
'path' is the result file name, None for standard output.  
'format'==1 iff the file is formatted with newlines for readability
```

matchone(self, pattern, atname)

```
# Return true iff the attribute with name atname is a string  
# attribute which matches the compiled regular expression pat  
# if atname is None and pattern matches at least one string  
# attribute. Return false if the attribute is not found or is not
```

searchone(self, pattern, atname)

```
Return true iff the attribute with name atname is a string  
attribute which contains the compiled regular expression patt  
if atname is None and pattern matches at least one string  
attribute. Return false if the attribute is not found or is not  
a string.
```

Methods inherited from [cdms.internattr.InternalAttributesClass](#):

is_internal_attribute(self, name)

`is internal attribute`(name) is true if name is internal.

replace_external_attributes(self, newAttributes)

`replace external attributes`(newAttributes)

Replace the external attributes with dictionary newAttributes

Methods inherited from [PropertiedClasses.Properties.PropertiedClass](#):

__delattr__(self, name)

__getattr__(self, name)

__setattr__(self, name, value)

get_property_d(self, name)

Return the 'del' property handler for name that self uses.

Returns None if no handler.

get_property_g(self, name)

Return the 'get' property handler for name that self uses.

Returns None if no handler.

get_property_s(self, name)

Return the 'set' property handler for name that self uses.

Returns None if no handler.

set_property(self, name, actg=None, acts=None, actd=None, nowrite=None, nodelete=None)

Set attribute handlers for name to methods actg, acts, actd

None means no change for that action.

nowrite = 1 prevents setting this attribute.

nowrite defaults to 0.

nodelete = 1 prevents deleting this attribute.

nodelete defaults to 1 unless actd given.

if nowrite and nodelete is None: nodelete = 1

Methods inherited from [cdms.cudsinterface.cuDataset](#):

__call__(self, id, *args, **kwargs)

Call a variable object with the given id. Exception if not fo

Call the variable with the other arguments.

__getitem__(self, key)

Implement f['varname'] for file/dataset f.

cleardefault(self)

Clear the default variable name.

default_variable(self, vname)
Set the default variable name.

dimensionarray(self, dname, vname=None)
Values of the dimension named dname.

dimensionobject(self, dname, vname=None)
CDMS axis object for the dimension named dname.

getattribute(self, vname, attribute)
Get the value of attribute for variable vname

getdimensionunits(self, dname, vname=None)
Get the units for the given dimension.

getglobal(self, attribute)
Get the value of the global attribute.

getslab(self, vname, *args, **keys)
getslab('name', arg1, arg2,) returns a cdms variable containing the data.

Arguments for each dimension can be:

- (1) : or None -- selected entire dimension
- (2) Ellipsis -- select entire dimensions between the ones
- (3) a pair of successive arguments giving an interval in world coordinates.
- (4) a cdms-style tuple of world coordinates e.g. (start, s

listall(self, vname=None, all=None)
Get info about data from the file.

listattribute(self, vname=None)
Get attributes of data from the file.

listdimension(self, vname=None)
Return a list of the dimension names associated with a variable. If no argument, return the file.axes.keys()

listglobal(self)
Returns a list of the global attributes in the file.

listvariable(self)
Return a list of the variables in the file.

listvariables = listvariable(self)
Return a list of the variables in the file.

readScripGrid(self, whichGrid='destination', checkGrid=1)
Read a SCRIP curvilinear or generic grid from the dataset. The dataset can be a SCRIP grid file or mapping file. If a map 'whichGrid' chooses the grid to read, either "source" or "des

If 'checkGrid' is 1 (default), the grid cells are checked for
and 'repaired' if necessary.
Returns the grid object.

showall(self, vname=None, all=None, device=None)
Show a full description of the variable.

showattribute(self, vname=None, device=None)
Show the attributes of vname.

showdimension(self, vname=None, device=None)
Show the dimension names associated with a variable.

showglobal(self, device=None)
Show the global attributes in the file.

showvariable(self, device=None)
Show the variables in the file.

class ***DuplicateAxisError***([cdms.error.CDMSError](#))

Method resolution order:

[DuplicateAxisError](#)
[cdms.error.CDMSError](#)
[exceptions.Exception](#)

Methods inherited from [cdms.error.CDMSError](#):

[__init__](#)(self, args='Unspecified error from package cdms')
[__str__](#)(self)

Methods inherited from [exceptions.Exception](#):

[__getitem__](#)(...)

Functions

createDataset(path, template=None)
Create a dataset
'path' is the XML file name, or netCDF filename for simple file
'template' is a string template for the datafile(s), for dataset

load(path)
Create a tree from a file path.
Returns the parse tree root node.

loadURI(uri)

```

# Create a tree from a URI
# URI is of the form scheme://netloc/path;parameters?query#fragment
# where fragment may be an XPointer.
# Returns the parse tree root node.

openDataset(uri, mode='r', template=None, dods=1)
    # Open an existing dataset
    # 'uri' is a Uniform Resource Identifier, referring to a cdunif fi
    #   or LDAP URL of a catalog dataset entry.
    # 'mode' is 'r', 'r+', 'a', or 'w'

parseFileMap(text)
    Parse a CDMS filemap. having the form:
    filemap ::= [ varmap, varmap, ...]
    varmap ::= [ namelist, slicelist ]
    namelist ::= [name, name, ...]
    slicelist ::= [indexlist, indexlist, ...]
    indexlist ::= [i,j,k,l,path]

parseIndexList(text)
    Parse a string of the form [i,j,k,l,path] where
    i,j,k,l are indices or '-', and path is a filename.
    Coerce the indices to integers, return (result, nconsumed).

parseName(text)

parseVarMap(text)
    Parse a string of the form [ namelist, slicelist ]

parselist(text, f)
    Parse a string of the form [A, A, ...].
    f is a function which parses A and returns (A, nconsumed)

```

Data

```

CdDatatypes = ['Char', 'Byte', 'Short', 'Int', 'Long', 'Float', 'Double', 'String']
DuplicateAxis = 'Axis already defined: '
DuplicateGrid = 'Grid already defined: '
DuplicateVariable = 'Variable already defined: '
FileNotFoundException = 'File not found: '
FileWasClosed = 'File was closed: '
InvalidDomain = 'Domain elements must be axes or grids'
ModeNotSupported = 'Mode not supported: '
SchemeNotSupported = 'Scheme not supported: '

```